IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Gregor P. Freund	Examiner: Ha, Leynna A
Serial No.: 10/605,189	Art Unit: 2188
Filed: September 12, 2003	APPEAL BRIEF
For: Security System with Methodology for Interprocess Communication Control	

Mail Stop Appeal
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

BRIEF ON BEHALF OF GREGOR P. FREUND

This is an appeal from the Final Rejection mailed 04/10/2007, in which the currently-pending claims stand finally rejected. Appellant filed a Notice of Appeal on 08-14-2007. This brief is submitted electronically in support of Appellant's appeal.

TABLE OF CONTENTS

1. REAL PARTY IN INTEREST

The real party in interest is assignee Check Point Software Technologies, Inc. located at 800 Bridge Parkway, Redwood City, CA 94065.

2. RELATED APPEALS AND INTERFERENCES

There are no appeals or interferences known to Appellant, the Appellant's legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

3. STATUS OF CLAIMS

The status of all claims in the proceeding is as follows:

Rejected: Claims 1-47

Allowed or Confirmed: None

Withdrawn: None

Objected to: None

Canceled: None

Identification of claims that are being appealed: Claims 1-47

An appendix setting forth the claims involved in the appeal is included as the last section of this brief.

4. STATUS OF AMENDMENTS

One Amendment has been filed in this case. Appellant mailed an Amendment on January 16, 2007, in response to a non-final Office Action dated October 16, 2006. In the Amendment, the pending claims were amended in a manner which Appellant believes clearly distinguished the claimed invention over the art of record, for overcoming the art rejections. In response to the Examiner's Final Rejection dated April 10, 2007, Appellant filed a Notice of Appeal. Appellant has chosen to forgo filing an Amendment After Final which might further limit Appellant's claims, as it is believed that further amendments to the claims are not warranted in view of the art. Accordingly, no Amendments have been entered in this case after the date of the Final Rejection.

5. SUMMARY OF CLAIMED SUBJECT MATTER

A. First Ground

As to Appellant's First Ground for appeal, Appellant asserts that the art rejection relying on Andrews fails to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention comprises the embodiment set forth in **independent claim 1**: a method for controlling interprocess communication (see, e.g., Appellant's specification at 300 (Fig. 3) and paragraphs [0064-0071], and 400 (Fig. 4) and paragraphs [0073-0082]), the method comprising: defining rules indicating which system services a given application can invoke using interprocess communication to invoke said system services (see, e.g., Appellant's specification glossary, definition of Security Policy at [0040]; see also rules engine described at paragraphs [0071] and [0078-0080]); trapping an attempt by a particular application to invoke a particular system service (see, e.g., Appellant's specification at paragraphs [0058-0059], [0067-0068], [0071], [0076-0077] and step 404 (Fig. 4)); identifying the particular application that is attempting to invoke the particular system service (see, e.g., Appellant's specification at [0077] which describes particular feature of Interprocess Communication Controller (IPCC) determining the currently running process or application which is attempting to open a connection); and based on identity of the particular application and on the rules indicating which system services a given application can invoke, blocking the attempt when the rules indicate that the particular application cannot invoke the particular system service (see, e.g., Appellant's specification at [0078] which describes particular feature of TrueVector engine (VS_MON), which includes a rules engine (or database) to determine whether or not to block the attempt by the specified application (e.g., the malware application) to open a channel to the target service (e.g., DNS service)).

Appellant further asserts that the art rejection relying on Andrews fails to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention comprises the embodiment set forth in **independent claim 14**: in a computer system, a method for regulating communications between processes (see, e.g., Appellant's specification at 300 (Fig. 3) and paragraphs [0064-0071], and 400 (Fig. 4) and paragraphs [0073-0082]), the method comprising: defining a policy specifying

whether one process may use interprocess communication to communicate with another process (see, e.g., Appellant's specification glossary, definition of Security Policy at [0040]; see also rules engine described at paragraphs [0071] and [0078-0080]); intercepting an attempt by a first process to communicate with a second process (see, e.g., Appellant's specification at paragraphs [0058-0059], [0067-0068], [0071], [0076-0077] and step 404 (Fig. 4)); identifying the first process that is attempting to communicate with the second process [and] identifying the second process (see, e.g., Appellant's specification at [0077] which describes particular feature of Interprocess Communication Controller (IPCC) determining the currently running process or application which is attempting to open a connection); based on said policy, determining whether the first process may communicate with the second process (see, e.g., Appellant's specification at [0078] which describes particular feature of TrueVector engine (VS_MON), which includes a rules engine (or database) to determine whether or not to block the attempt by the specified application (e.g., the malware application) to open a channel to the target service (e.g., DNS service)); and allowing the first process to communicate with the second process if said policy indicates that the first process may communicate with the second process (see, e.g., Appellant's specification at aforementioned [0078]; also refer to [0071] where IPCC's allowing and blocking of messages is described).

Appellant further asserts that the art rejection relying on Andrews fails to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention comprises the embodiment set forth in **independent claim 25**: a method for controlling interprocess communications from one application to another (see, e.g., Appellant's specification at 300 (Fig. 3) and paragraphs [0064-0071], and 400 (Fig. 4) and paragraphs [0073-0082]), the method comprising: registering a first application to be protected from interprocess communications of other applications (see, e.g., Appellant's specification glossary, definition of Security Policy at [0040]; see also rules engine described at paragraphs [0071] and [0078-0080]); detecting an attempt to access the first application using interprocess communication (see, e.g., Appellant's specification at paragraphs [0058-0059], [0067-0068], [0071], [0076-0077] and step 404 (Fig. 4)); identifying a second application that is attempting to access the first application using

interprocess communication (see, e.g., Appellant's specification at [0077] which describes particular feature of Interprocess Communication Controller (IPCC) determining the currently running process or application which is attempting to open a connection); and rerouting the attempt to access the first application through an interprocess communication controller that determines whether to allow the attempt, based on rules indicating whether the second application may access the first application using interprocess communication (see, e.g., Appellant's specification at [0078] which describes particular feature of TrueVector engine (VS_MON), which includes a rules engine (or database) to determine whether or not to block the attempt by the specified application (e.g., the malware application) to open a channel to the target service (e.g., DNS service)).

Appellant further asserts that the art rejection relying on Andrews fails to teach or suggest all of the claim limitations of Appellant's claimed invention, where the claimed invention comprises the embodiment set forth in **independent claim 36**: a system for regulating interprocess communication between applications (see, e.g., Appellant's specification at 300 (Fig. 3) and paragraphs [0064-0071], and 400 (Fig. 4) and paragraphs [0073-0082]), the system comprising: a policy specifying applications that are permitted to communicate with a first application using interprocess communication (see, e.g., Appellant's specification glossary, definition of Security Policy at [0040]; see also rules engine described at paragraphs [0071] and [0078-0080]); a module for detecting a second application attempting to communicate with the first application using interprocess communication (see, e.g., Appellant's specification at paragraphs [0058-0059], [0067-0068], [0071], [0076-0077] and step 404 (Fig. 4)); and an interprocess communication controller for identifying the second application attempting to communicate with the first application and determining whether to permit the communication based upon the identification of the second application and the policy specifying applications permitted to communicate with the first application (see, e.g., Appellant's specification at [0077] which describes particular feature of Interprocess Communication Controller (IPCC) determining the currently running process or application which is attempting to open a connection; see also, e.g., Appellant's specification at [0078] which describes particular feature of TrueVector engine (VS_MON), which includes a rules engine (or database) to

determine whether or not to block the attempt by the specified application (e.g., the malware application) to open a channel to the target service (e.g., DNS service)).

6. GROUNDS OF REJECTION TO BE REVIEWED

The grounds presented on appeal are:

(1st) Whether claims 1-47 are unpatentable under 35 ·U.S.C. 102(e) as being anticipated by Andrews (US 6,574,736).

7. ARGUMENT

A. First (and single) Ground: Claims 1-47 rejected under Section 102(e)

1. General

Under Section 102, a claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in the single prior art reference. (See, e.g., MPEP Section 2131.) As will be shown below, the reference fails to teach each and every element set forth in Appellant's independent claims, as well as other claims, and therefore fails to establish anticipation of the claimed invention under Section 102.

2. Claims 1-47

Claims 1-47 stand rejected under 35 ·U.S.C. 102(e) as being anticipated by Andrews (US 6,574,736). The Examiner's rejection of claim 1 is representative:

> Andrews discloses a method for controlling interprocess communication, the method comprising:
>
> defining rules indicating which system services a given application can invoke (col. 9, lines 34-38 and 49-55; Andrews disclose object oriented programming, programs are written as a collection of object class which each model real world or abstract items by combining data to represent the item's properties with methods (e.g., program functions or procedures) to represent the item's functionality. An object is an instance of a programmer defined type referred to as a class, which exhibits the

characteristics of data encapsulation, polymorphism, and inheritance (col. 2, lines 1-10). The term application can also broadly give in light as a program, object, software, routine, or module (col. 6, lines 23-26). With this in mind, Andrews discloses application role is a role defined of the application to define access privileges to processing services to the applications. More than two roles can be associated, and the roles maybe from the same or different applications (col. 4, lines 27-28). In addition, the security framework including role definitions, to determine whether access to a component's functionality is permitted where the framework blocks calls to objects if access is not permitted (col. 4, lines 47-53). Thus, the rules refer to the roles of the applications in addition to being populated with users or the types of users able 'to access the applications (col. 4, lines 25-35). Thus, Andrews reads on the claimed defining rules for the applications.)

using interprocess communication to invoke said system services; (col. 16, lines 16-23 and co1. 21, lines 36-53; the interprocess communication is the ability of a task or process to communicate with another in a multitasking operating system (col. 1, lines 50-54 and col. 2, lines 47-50). Hence, an object being executed to communicate via signals (col. 8, lines 1-15) with another to perform different functions (e.g. defining and regulating rules or transferring processes) or tasks being performed would read on the claimed interprocess communications (col. 6, lines 34-40 and col. 8, lines 43-51). Andrews discloses launching or executing a program where the operating system associates the user id with the process in which the program is run and with the process' threads, When a thread executing on the user's behalf then accesses a system resource, the operating system performs an authorization check to verify the user. Andrews further teaches the present invention is directed toward a method and system for providing an object execution environment with a security framework providing automatic security services for composable applications (col. 5, lines 63-67). Thus, Andrews reads on the

claimed using interprocess communication to invoke system services (col. 19, lines 15-30).)

trapping an attempt by a particular application to invoke a particular system service; (col.15, lines 20-22 and col.21, lines 35-37)

identifying the particular application that is attempting to invoke the particular system service; and (col.20, lines 15-16 and col.22, lines 13-18 and 30-32)

based on identity of the particular application and on the rules indicating which system services a given application can invoke (col. 14, lines 56-59),

blocking the attempt when the rules indicate that the particular application cannot invoke the particular system service. (col. 22, lines 1-9).

For the reasons set forth below, Appellant's claimed invention may be distinguished on a variety of grounds.

Appellant's invention and patent claims are directed to controlling access of potentially "bad" applications or processes, such as "malware" inadvertently downloaded from the Internet, that may surreptitiously invoke operating system services (e.g., Microsoft Windows DNS service) using interprocess communication (IPC) technique for compromising a user's computer. Prior art computer systems have classically adopted user-based approaches to controlling security -- that is, by assuming that if "bad" (i.e., unauthorized) users are denied access to operating programs then computer system security is achieved. Malware attacks however do not involve "bad" users gaining access but, instead, involve "good" (i.e., authorized) users accidentally executing "bad" programs which in turn attack the computer system (e.g., for gaining access to sensitive information). For example, a "good" user may be duped into executing a malware program by clicking on (launching) an e-mail attachment. More recently, computer systems have attempted to improve security beyond simple user-based approaches by identifying and blocking the "bad" programs (i.e., malware) themselves. Hackers and

malware purveyors have not sit still, however, but instead have taken additional steps to effectively cloak malware from such detection. Malware attacks using IPC are particularly insidious to detect, because the malware (i.e., "bad") program hides its activity by running it through what is normally a safe and "good" operating system service or other "good" program (i.e., previously established to not be malware). Prior art program-based security systems do not detect or prevent this type of attack, which relies on IPC invocation of "good" (friendly) programs or processes, especially operating system services.

In Appellant's last-filed Amendment, Appellant's claims were amended to highlight the specific features of Appellant's invention that address the above vulnerability that is posed by interprocess communication, for instance, that one application or process may attempt to use interprocess communication to invoke operating system services in a manner that thwarts security measures. Importantly, security is not based on the identity of a particular user or his/her role or authorization, or even on the identification of a particular "bad" program, but is instead based on the notion that certain operating system services or other "good" processes are vulnerable to invocation by malicious programs, for example allowing the operating system to be tricked into carrying out the malicious or compromising act. Vulnerable operating system services in particular should be protected from IPC invocation by malicious programs, regardless of how safe or authorized the currently logged-in user is thought to be.

In response to Appellant's last-filed Amendment, the Examiner has revamped his argument by extensively remapping Appellant's claim limitations into the cited art, with the result that the Examiner's position is now less discernable. In any case, the cited art of Andrews simply does not function in the same manner as Appellant's claimed invention. The Examiner's remapping of Appellant's claim limitations into multiple disparate sections of Andrews cannot serve as a basis to impart functionality to Andrews' system that it clearly lacks.

Andrews' teaching is directed to "composable roles" -- that is, defining access privileges for users. As described by Andrews' Abstract, in Andrews's system an application developer grants access privileges to application processing services in an

object-based application by defining logical classes of users called "roles." When the application is deployed on a host computer system, an administrator populates the roles with users and groups (of users) recognized by the host computer system. At runtime, a user is not permitted access to a processing service unless the user is a member of a permitted role for the processing service. To ease administration, two or more roles can be composed. In one implementation, roles are associated with a separate composite role. The administrator can then populate the composite role instead of individually populating each of the roles associated with the composite role.

The foregoing approach of Andrews has little in common with Appellant's invention. Significantly, if an Andrews' computer system were inadvertently infected with malware by an authorized user (e.g., an authorized user duped into clicking on an e-mail attachment), there simply is no facility or means described for Andrews's system to prevent the malware from executing and potentially compromising the computer (e.g., gaining access to sensitive data or even gaining control over the entire computer system). Instead, Andrews' system would look to the user and his or her role -- as that is what Andrews explicitly describes should be done -- in order to determine what security measures to apply. In such a case even with a "good" or authorized user, the Andrews' system simply cannot correctly determine that the "good" or authorized user is in fact inadvertently running a "bad" or unauthorized malware program. To contend otherwise, the Examiner ignores the plain teaching of Andrews and instead embellishes Andrews with hindsight benefits lifted from Appellant's specification.

Andrews describes in detail the problem that he is addressing and which developers face: finding a way to control user access (i.e., implement a user-based security scheme) efficiently in object-oriented applications. For example, a developer of a banking application may want to prevent tellers (i.e., specific group of users) from accessing an object for changing a customer's deposit balance. However, the developer may be designing an application for use by a wide variety of banking organizations with different organizational structures and different employees (again, note that this is user-based control). It would be a burdensome task for the application developer to customize the banking application for each of the banking organizations. Andrews emphasizes that if user identities specific to the banking organization were placed in the objects' logic,

maintaining the application would require the further burden of changing the logic to account for employee turnover or reorganization. (See, e.g., Andrews at column 2, lines 31-44). His particular focus is on controlling **user access** in the context of object-oriented applications, where functionality may be divvied up among different components, such as an "extended application" having separate shipping and billing components. In the extended application, the developer of the shipping component may have defined a user role of "manager" while the developer of the billing component may have defined a role of "mgr". The particular solution provided by Andrews to this problem is to apply his "composing roles." An administrator can associate a first role with a second role, which takes on the population of the first role. Subsequent changes to the first role's population are automatically implemented to the second role's population. Clearly, Andrews' teaching is directed to controlling user access -- that is, what humans are doing. Andrews has no description or notion that there may be rogue programs being executed by an authorized user.

Appellant's invention provides a security system with methodology for controlling applications use of interprocess communication (IPC, communications that may occur between processes that operate on computer systems), totally regardless or irrespective of who the user is or what he or she is doing. By focusing on the applications and the processes or services that they access, instead of focusing on users, Appellant's invention can thwart rogue applications or malware, even those applications that can fool systems such as Andrews by running with the (apparent) permission of an authorized user.

IPC poses a particularly difficult problem for security management, as a "bad" application may mask an action by requesting (via IPC) a "good" process or system service to perform the action. For example, the "bad" application (malware) may trick the computer system into connecting to a malicious server by having the "bad" application invoke the operating system's DNS service ("good" service, of the operating system). Here, the "bad" application effectively "flies below the radar" of the computer and its (prior art) security systems, because those systems only see the "good" service performing what is assumed to be an innocent action. IPC occurs directly between processes (e.g., between an application and a service) without any involvement of the user. In fact, the user will typically be a bona fide or fully authorized user. Importantly,

the specific vulnerability addressed by Appellant's invention does not even involve the user, but instead involves a malicious process (malware) that has been placed on the user's machine. In this scenario, the user's own privileges (access rights or "role") are not at issue: the user is assumed to be a bona fide user with appropriate authority to use his or her computer. Instead, the issue addressed by Appellant's invention is that the user has unfortunately picked up malware, such as by accidentally downloading it from the Internet (e.g., by well known conduits, such as Trojan horse programs, viruses, worms, and the like), and the malware itself is able to cloak its activities by running them through services or processes thought to be good. Andrews, in contrast, is concerned about controlling users for preventing an unauthorized user (i.e., a human) from gaining access to information or resources that he or she is not authorized for. Given those fundamental differences, a thorough review of Andrews confirms that it is directed to an aspect of computer security that is completely different from Appellant's.

To be sure, Appellant's invention shares some features with Andrews' system as both are in the field of computer security; both share general features with other computer security products as well. For example, both may employ security rules for defining conditions. Also, both may block a given access attempt, if an applicable rule or condition is not met. However, apart from general security features (present in all security-related solutions), Appellant's claimed invention includes specific features pertaining to controlling applications that invoke interprocess communication that may occur between separate processes completely irrespective of the user or his or her access privileges. These features are not taught or suggested by Andrews. In order to highlight this distinction, Appellant's independent claims were amended in Appellant's last-filed Amendment to explicitly recite the controlling of applications (irrespective of users) that employ interprocess communication to invoke other processes (e.g., especially system services, as in the case of independent claim 1).

Turning now to the specific teachings of Andrews cited by the Examiner, one finds that the specific points of Andrews highlighted by the Examiner bears little resemblance to Appellant's claim limitations. The Examiner's scattershot or randomly inclusive approach to piecing together disparate bits of Andrews demonstrate, if anything, the tenuous nature of the rejection. For example, for Appellant's first claim

limitation of "defining rules indicating which system services a given application can invoke using interprocess communication to invoke said system services," the Examiner loosely weaves together 16 disparate citations, jumping through Andrews in a hopscotch-like manner: 1: col. 9, lines 34-38; 2: col. 9, lines 49-55; 3: col. 2, lines 1-10; 4: col. 6, lines 23-26; 5: col. 4, lines 27-28; 6: col. 4, lines 47-53; 7: col. 4, lines 25-35; 8: col. 16, lines 16-23; 9: col. 21, lines 36-53; 10: col. 1, lines 50-54; 11: col. 2, lines 47-50; 12: col. 8, lines 1-15; 13: col. 6, lines 34-40; 14: col. 8, lines 43-51; 15: col. 5, lines 63-67; and 16: col. 19, lines 15-30.

In any event, the crux of the Examiner's rejections appears to be "the rules refer to the roles of the applications in addition to being populated with users or the types of users able to access the applications (col. 4, lines 25-35)" (Examiner's Final Rejection, Paragraph 3). Here, the Examiner latches on to the phrase "application roles" to assert that Andrews also has a system for controlling applications in a manner similar Appellant's invention (i.e., independent of users). However, this is not what Andrews himself contends. The particular section of Andrews cited by the Examiner in support of the foregoing proposition instead reads:

> SUMMARY OF THE INVENTION
>
> The present invention includes a method and system for composing roles. An administrator can associate a first role with a second role, which takes on the population of the first role. Subsequent changes to the first role's population are automatically implemented to the second role's population. More than two roles can be associated, and the roles may be from the same or different applications. As a result, an administrator can perform one operation to bind a user to multiple roles.
>
> In one aspect of the invention, the roles are bound to composite roles. The binding can be one to one, many to one, or one to many. The composite roles are populated with users, and users populating a composite role are considered to be members of the roles bound to the composite role. In this way, an administrator can populate a single composite role instead of individually populating separate application roles.

(Andrews col. 4, lines 25-35, with the addition of lines 20-24 (first two sentences))

As shown by the above quote, it is clear that Andrews' particular focus is on

controlling **user access**. And Andrews' particular innovation is that this is being done in the context of object-oriented applications, especially "extended applications" (e.g., having separate shipping and billing components). Andrews defines a role as follows: "A role is a logical class of **users** having access privileges to processing services of an application." (Andrews at col. 9, lines 34-35, emphasis added.) Importantly, an "application role" is a logical class of **users** defined at an application's development time as having access privileges to processing services of the application. Andrews' "application roles" is not a system for controlling applications invocation of operating system services in the manner of Appellant's invention, which (among other things) operates to control malware irrespective of users.

Andrews true focus and teaching is the creation of a "composite role," where "a composite role is a role bound to at least one role," and "a role is a logical class of users having access privileges to processing services of an application." Andrews' "composite role" may include an "application role," but note that an application role is simply a user role bound to a particular application at development time. For example, Andrews states the example: "Since the 'tellers' role and the 'managers' role are associated with a particular application and are defined as development of the application they are sometimes called 'application roles.'" (Andrews at column 9, lines 62-65.) Here, Andrews is concerned about particular functionality of an application, such as "an object method for changing a customer's address" (see Andrews at column 9, lines 56-57). Andrews does not discuss interprocess communication between different processes (e.g., between two applications, or between an application and a system service), nor would one expect to find such a discussion since Andrews is focused on controlling access of users. Andrews is concerned about an unauthorized or malicious human user gaining inappropriate access to sensitive computing services. Andrews is completely unconcerned about the scenario of an authorized user whose computer has been infected with malware that is attempting unauthorized interprocess communication as a means to mask its malicious activity.

In order to further highlight this distinction of Appellant's invention, Appellant's claim 1 was amended in Appellant's last-filed Amendment to recite (shown in amended form):

defining rules indicating which system services a given application can invoke using interprocess communication to invoke said system services;

Here, the concern being addressed is that a given application -- running on an authorized user's computer -- may potentially use interprocess communication to invoke other processes, such as system services, in an inappropriate manner. These are "bad acts" that may occur wholly between computer processes irrespective of the user -- that is, irrespective of any user status, role, or even user awareness of the situation. The user himself or herself is assumed to be a fully authorized user and thus is not the threat being addressed (unlike Andrews, who is essentially describing a mechanism for keeping out bad (unauthorized) human users from accessing particular services of an application). Appellant's other independent claims include similar claim limitations highlighting these distinguishing features.

Regarding Appellant's claim limitation of "trapping an attempt by a particular application to invoke a particular system service," the Examiner points to Andrews at column 15, lines 20-22 and column 21, lines 35-37. In the field of computer science, many scenarios arise where something needs to be trapped or intercepted, and Appellant certainly makes no claim to have invented the notion by itself. In the above section cited by the Examiner, however, note particularly that this intercepting or trapping described by Andrews is specific user-based intercepting. Andrews explains: "To indicate which user initiated execution of the objects, an identity is associated with calls from the client object 706 (e.g., the identity of a logged on user or an identity indicating the system user) as described in more detail below." (Andrews at column 15, lines 28-32). Clearly, Andrews is describing a scenario of a user attempting to initiate execution of objects. Andrews is totally devoid of any description relating to detecting and controlling an application process -- not necessarily operating under control or knowledge of the user, such as malware -- that attempts to initiate or invoke another process's services as a means of concealing malicious activity.

Further, regarding Appellant's limitation of "blocking the attempt when the rules

indicate **that the particular application cannot invoke** the particular system service," the Examiner points to Andrews at column 14, lines 56-59 and column 22, lines 1-9. However, one finds that Andrews describes: "At run time, a security service monitors calls to objects and limits access to those **user** identities that are members of the role associated with the method, interface, object, or application being called." (Emphasis added.) Significantly, Andrews has no notion that a particular application (e.g., malware) itself may be unauthorized, and therefore may need its access to sensitive system services controlled. (Not surprisingly, a text search of Andrews' patent reveals **absolutely no** discussion whatsoever of "malware", "viruses", "Trojan horses", "worms", or the like, nor does Andrews specification even mention "interprocess communication".) Andrews instead focuses his teachings on controlling user access, and as a result fails to provide sufficient teaching or suggestion to anticipate Appellant's claimed invention.

All told, Andrews' teaching is directed to controlling user access -- that is, what humans are doing. Andrews has no description or notion that a rogue program may attempt to invoke another (innocent) process to masquerade on its behalf (such as, duping the operating system to post sensitive information to a hacker's DNS server). Andrews does not even have any discussion about a rogue program being accidentally or inadvertently executed by a user, authorized or not. Appellant's claims set forth a patentable advance in the area of controlling access of potentially "bad" applications that may cloak their malicious activities through interprocess communication. Appellant's claims have been amended to highlight the specific features of Appellant's invention that address the vulnerability posed by interprocess communication, that one application or process may attempt to use interprocess communication to thwart security measures by running its malicious activities through what is considered to be a "good" (non-malware) process or service, thereby hiding its activities from prior art security measures (e.g., prior art firewalls). Accordingly, it is respectfully submitted that the claims distinguish over Andrews and the Examiner's rejection under Section 102 should not be sustained.

B. Conclusion

The present invention greatly improves the ease and efficiency of the task of

providing security for vulnerable computer system services, which are susceptible to invocation from malware via interprocess communication. It is respectfully submitted that the present invention, as set forth in the pending claims, sets forth a patentable advance over the art.

In view of the above, it is respectfully submitted that the Examiner's rejections under 35 U.S.C. Section 102 not be sustained. If needed, Appellant's undersigned attorney can be reached at 408 884 1507. For the fee due for this Appeal Brief, please refer to the attached Fee Transmittal Sheet. This Brief is submitted electronically.

Respectfully submitted,

Date: November 10, 2007

/John A. Smart/

John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX

8. CLAIMS APPENDIX

1. A method for controlling interprocess communication, the method comprising:

defining rules indicating which system services a given application can invoke using interprocess communication to invoke said system services;

trapping an attempt by a particular application to invoke a particular system service;

identifying the particular application that is attempting to invoke the particular system service; and

based on identity of the particular application and on the rules indicating which system services a given application can invoke, blocking the attempt when the rules indicate that the particular application cannot invoke the particular system service.

2. The method of claim 1, wherein said trapping step includes intercepting operating system calls for invoking the particular system service.

3. The method of claim 1, wherein said trapping step includes intercepting local procedure calls for invoking the particular system service.

4. The method of claim 1, wherein said trapping step includes intercepting an attempt to open a communication channel to the particular system service.

5. The method of claim 1, wherein said trapping step includes rerouting an attempt to invoke the particular system service from a system dispatch table to an interprocess communication controller for determining whether to block the attempt based on the rules.

6. The method of claim 5, wherein said step of rerouting attempts to invoke the particular system service from a dispatch table to the interprocess communication controller includes replacing an original destination address in the system dispatch table with an address of the interprocess communication controller.

7. The method of claim 6, further comprising the steps of:

retaining the original destination address; and

using the original destination address for invoking the particular system service if the interprocess communication controller determines not to block the attempt.

8. The method of claim 1, wherein the rules specifying which system services a given application can invoke are established based on user input.

9. The method of claim 1, wherein the step of blocking the attempt is based upon consulting a rules engine for determining whether the particular application can invoke the particular system service.

10. The method of claim 1, wherein the step of blocking the attempt includes obtaining user input as to whether the particular application can invoke the particular system service.

11. The method of claim 10, wherein said step of obtaining user input as to whether the particular application can invoke the particular system service includes the substeps of:

providing information to the user about the particular application that is attempting to invoke the particular system service; and

receiving user input as to whether the particular application should be blocked from invoking the particular system service.

12. A computer-readable medium having computer-executable instructions for performing the method of claim 1.

13. The method of claim 1, further comprising:

downloading a set of computer-executable instructions for performing the method of claim 1.

14. In a computer system, a method for regulating communications between processes, the method comprising:

defining a policy specifying whether one process may use interprocess communication to communicate with another process;

intercepting an attempt by a first process to communicate with a second process;

identifying the first process that is attempting to communicate with the second process;

identifying the second process;

based on said policy, determining whether the first process may communicate with the second process; and

allowing the first process to communicate with the second process if said policy indicates that the first process may communicate with the second process.

15. The method of claim 14, wherein the first process comprises an instance of an application program.

16. The method of claim 14, wherein the second process comprises a system service.

17. The method of claim 14, wherein said intercepting step includes intercepting operating system calls made by the first process to attempt to communicate with the second process.

18. The method of claim 14, wherein said intercepting step includes detecting local procedure calls.

19. The method of claim 14, wherein said intercepting step includes detecting an attempt by the first process to open a communication channel to the second process.

20. The method of claim 14, wherein said intercepting step includes rerouting attempts by the first process to communicate with the second process from a system

dispatch table to an interprocess communication controller.

21. The method of claim 14, wherein said step of identifying the second process includes evaluating parameters of the attempt made by the first process to communicate with the second process.

22. The method of claim 14, wherein said policy specifies particular processes to be protected from communications made by other processes.

23. The method of claim 14, further comprising:

providing for a process to be registered in order to be protected from communications made by other processes; and

determining whether to allow the first process to communicate with the second process based, at least in part, upon determining whether the second process is registered.

24. The method of claim 23, wherein said determining step is based, at least in part, on the type of communication the first process is attempting with the second process.

25. A method for controlling interprocess communications from one application to another, the method comprising:

registering a first application to be protected from interprocess communications of other applications;

detecting an attempt to access the first application using interprocess communication;

identifying a second application that is attempting to access the first application using interprocess communication; and

rerouting the attempt to access the first application through an interprocess communication controller that determines whether to allow the attempt, based on rules indicating whether the second application may access the first application using interprocess communication.

26. The method of claim 25, wherein said registering step includes supplying rules specifying particular communications from which the first application is to be protected.

27. The method of claim 26, wherein the interprocess communication controller determines whether to allow the attempt based, at least in part, upon the rules specifying particular communications from which the first application is to be protected.

28. The method of claim 25, wherein said detecting step includes intercepting operating system calls for accessing the first application.

29. The method of claim 25, wherein said detecting step includes detecting a graphical device interface (GDI) message sent to the first application.

30. The method of claim 29, wherein said identifying step includes evaluating parameters of the message sent to the first application.

31. The method of claim 25, wherein said detecting step includes detecting an attempt to send keystoke data to a window of the first application.

32. The method of claim 25, wherein said detecting step includes detecting an attempt to send mouse movement data to a window of the first application.

33. The method of claim 25, wherein said rerouting step includes rerouting the attempt to access the first application from a system dispatch table to the interprocess communication controller.

34. The method of claim 25, wherein said rules indicating whether the second application may access the first application includes rules indicating particular types of communications which are allowed.

35. The method of claim 25, further comprising:

if the interprocess communication controller allows the attempt to access the first application, routing the attempt to the first application.

36. A system for regulating interprocess communication between applications, the system comprising:

a policy specifying applications that are permitted to communicate with a first application using interprocess communication;

a module for detecting a second application attempting to communicate with the first application using interprocess communication; and

an interprocess communication controller for identifying the second application attempting to communicate with the first application and determining whether to permit the communication based upon the identification of the second application and the policy specifying applications permitted to communicate with the first application.

37. The system of claim 36, wherein said policy includes rules indicating particular types of communications which are permitted.

38. The system of claim 36, further comprising:

a rules engine for specifying applications that are permitted to communicate with the first application using interprocess communication.

39. The system of claim 36, further comprising:

a registration module for establishing said policy.

40. The system of claim 39, wherein said registration module provides for identifying applications to be governed by said policy.

41. The system of claim 36, wherein said module for detecting a second application detects an operating system call to open a communication channel to the first

application.

42. The system of claim 36, wherein said module for detecting a second application detects a graphical device interface (GDI) message sent to the first application.

43. The system of claim 36, wherein said module for detecting a second application detects a local procedure call attempting to access the first application.

44. The system of claim 36, wherein said module for detecting a second application redirects attempts to communicate with the first application to the interprocess communication controller.

45. The system of claim 36, wherein said module for detecting a second application reroutes the attempt to communicate with the first application from a dispatch table to the interprocess communication controller.

46. The system of claim 36, wherein said interprocess communication controller determines whether to permit the communication based, at least in part, upon evaluating parameters of the attempt made by the second application to communicate with the first application.

47. The system of claim 36, wherein said interprocess communication controller determines whether to permit the communication based upon obtaining user input as to whether to permit the second application to communicate with the first application.

9. EVIDENCE APPENDIX

This Appeal Brief is not accompanied by an evidence submission under §§ 1.130, 1.131, or 1.132.

10. RELATED PROCEEDINGS APPENDIX

Pursuant to Appellant's statement under Section 2, this Appeal Brief is not accompanied by any copies of decisions.